Ann Math Artif Intell (2011) 62:187–217
DOI 10.1007/s10472-011-9265-7

Experimental evaluation of pheromone models in ACOPlan

Marco Baioletti · Alfredo Milani · Valentina Poggioni · Fabio Rossi

Published online: 26 August 2011

Abstract In this paper the system ACOPlan for planning with non uniform action cost is introduced and analyzed. ACOPlan is a planner based on the ant colony optimization framework, in which a colony of planning ants searches for near optimal solution plans with respect to an overall plan cost metric. This approach is motivated by the strong similarity between the process used by artificial ants to build solutions and the methods used by state–based planners to search solution plans. Planning ants perform a stochastic and heuristic based search by interacting through a pheromone model. The proposed heuristic and pheromone models are presented and compared through systematic experiments on benchmark planning domains. Experiments are also provided to compare the quality of ACOPlan solution plans with respect to state of the art satisficing planners. The analysis of the results confirm the good performance of the Action–Action pheromone model and points out the promising performance of the novel Fuzzy–Level–Action pheromone model. The analysis also suggests general principles for designing performant pheromone models for planning and further extensions of ACOPlan to other optimization models.

Keywords Automated planning · Ant colony optimization

Mathematics Subject Classifications (2010) 68T05 · 68T20 · 68W20 · 68W25

M. Baioletti · A. Milani · V. Poggioni (✉) · F. Rossi
Department of Mathematics and Computer Science, University of Perugia,
Via Vanvitelli 1, 06123 Perugia, Italy
e-mail: poggioni@dmi.unipg.it

M. Baioletti
e-mail: baioletti@dmi.unipg.it

A. Milani
e-mail: milani@dmi.unipg.it

F. Rossi
e-mail: rossi@dmi.unipg.it

1 Introduction

Planning optimization relies on finding solution plans of highest quality with respect to a given objective function.

Most propositional planners embed basic optimization metrics like *sequential plan length*, or the number of *time steps* when parallel actions are allowed. HSP [8] is an example of planners of the first type; while many others, including GraphPlan [6] and BlackBox [21], optimize the number of steps. However planning frameworks in which optimization seems to be more natural are planning models with *action costs* and/or *numerical fluents*. In the first case a typical optimization metric is the overall plan cost, computed as the sum of costs of the actions occurring in the solution plan. On the other hand, numerical fluents are used to model resources, numerical constraints and other numerical aspects of the domain; in this case the metric is a function of numerical variables, like for instance the fuel consumption in the classical *Logistics* domain.

The main idea of this paper is to use the well known *Ant Colony Optimization* metaheuristic (*ACO*) [13] to solve planning problems with the aim of optimizing the quality of the solution plans. The approach is based on the strong similarity between the process used by artificial ants to build solutions and the way used by state—based planners to find solution plans. Therefore, we have defined an ACO algorithm which handles a colony of planning ants with the purpose of solving planning problems by optimizing solution plans with respect to the overall plan cost. According to the main features of ACO the ant–planners of the colony are stochastic and heuristic–based.

ACO is a metaheuristic inspired by the behavior of natural ants colony which has been successfully applied to many *Combinatorial Optimization* problems. Being ACO a stochastic incomplete algorithm, there is no guarantee that optimal solutions are ever found, but a number of successful applications confirm ACO as a generally promising metaheuristic to find efficiently and effectively good suboptimal solutions.

In order to investigate if ACO can be effectively used in optimization planning problems, we have implemented and tested *ACOPlan*, an ACO—based propositional planner which tries to optimize plans in terms of plan length. The promising results of *ACOPlan* [2–5] have motivated the extension to more complex planning frameworks, like planning with action costs. In this paper the *ACOPlan* model for planning with action costs is presented together with a systematic experimental evaluation of the system and a comparison with respect to the most recent benchmark planners for planning with action costs [18]. The evaluation focuses in particular on the role of different pheromone models on the performance in the benchmark domains.

The paper is structured as follows. Two brief introductions to Ant Colony Optimization and Automated Planning are provided, then the ACOPlan model and algorithm in the framework of planning with action cost are presented. Four pheromone models are introduced and motivated. Systematic experiments of the proposed pheromone models on benchmark domains are presented and the evaluation of their performance with respect to the other state of art planners is presented and discussed. The paper concludes with a discussion of lessons learned and future direction of works and extensions.

2 Ant Colony Optimization

Ant Colony Optimization is a metaheuristic used to solve combinatorial optimization problems, introduced since early 1990s by Dorigo and Stuetzle [13].

ACO is inspired by the foraging behavior of natural ant colonies that, when walking, release on the ground a chemical substance called *pheromone* that other ants can smell; moreover, ants tend to follow paths with a higher amount of pheromone. When an ant finds some food, it comes back to the nest releasing a higher quantity of pheromone with the implicit aim of communicating to other ants where the food is located. Pheromone trails are also subject to evaporation. The reduction of the pheromone quantities has a greater impact on the longest paths, because they would need to have more time to be enforced.

On the long run, the shortest paths from the nest to the food become marked by a higher quantity of pheromone since they are followed by an increasing number of ants.

ACO is usually applied to optimization problems whose solutions are composed by discrete components. A Combinatorial Optimization problem is described in terms of a *solution space* S, a set of (possibly empty) *constraints* Ω and an *objective function* $f : S \to \mathbb{R}^+$ to be minimized (maximized).

ACO uses a colony of artificial ants, which move in the solution space S and build solutions by composing discrete components. The choice of the component structure is crucial and it is strongly dependent on the class of problems we are solving. The construction of a solution is incremental: each ant probabilistically chooses a component to add to the partial solution built so far, according to the problem constraints Ω. Each component should contain all the information that is necessary to the ants to make a choice. The random choice is biased by the *artificial* pheromone value τ related to each component and by a heuristic function η. Both terms evaluate the desiderability of each component. The probability that an ant will choose the component c is

$$p(c) = \frac{[\tau(c)]^\alpha [\eta(c)]^\beta}{\sum_x [\tau(x)]^\alpha [\eta(x)]^\beta} \tag{1}$$

where the sum on x ranges on all the components which can be chosen, and α and β are tuning parameters which differentiate the pheromone and heuristic contributions.

The heuristic function is an external source of information and helps the ants to choose the components. Even if it has no equivalent in natural ants, it is an essential tool in ACO, because otherwise the artificial ants would be not able to solve significant problems.

The pheromone values represent a kind of memory shared by the whole ant colony and are subject to *u*pdate and *e*vaporation. In particular, the pheromone can be updated at each construction step (*o*nline update) or after complete solutions are created (*o*ffline update). Often only the best solutions are considered for the update: the best solution found so far (*best–so–far*) or best solution found in the current iteration (*iteration–best*) or both. In certain variants of ACO, limits for maximum and minimum pheromone values are used. Most ACO algorithms use the following update rule [7]:

$$\tau(c) \leftarrow (1-\rho) \cdot \tau(c) + \rho \cdot \sum_{s \in \Psi_{upd} : c \in s} F(s) \tag{2}$$

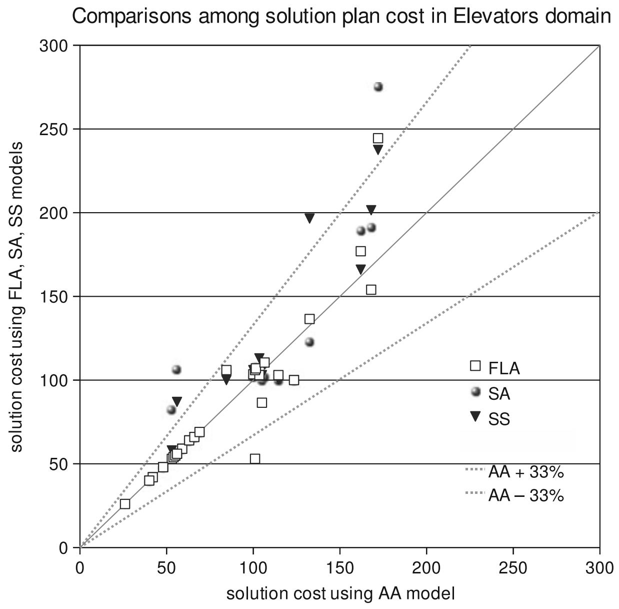

Fig. 1 Scatter plot comparing the solutions of the AA model versus the other models in terms of solution cost in the *Elevators* domain

where Ψ_{upd} is the set of solutions involved in the update, F is the so called *quality function*, which is a decreasing function of f (increasing if f is to be maximized), and ρ is the pheromone evaporation rate $\rho \in]0, 1[$. This pheromone evaporation rate allows to model the actual pheromone evaporation event and has been introduced in order to avoid a premature convergence towards non optimal solutions.

The simulation of the ant colony is iterated until a satisfactory solution is found, a termination criterion is satisfied or a given number of iterations is reached. The standard ACO algorithm is described in Fig. 1 where n_a is the number of the ants, θ_{iter} is the solution built by each ant and $best$ is the best solution found so far.

Algorithm 1 A classical ACO algorithm

```
1:  InitializePheromone($T$)
2:  $best \leftarrow$ NULL
3:  while not termination criterion do
4:      $\theta_{iter} \leftarrow \emptyset$
5:      for $i \leftarrow 1$ to $n_a$ do
6:          $\sigma \leftarrow$ BuildSolution($T$)
7:          $\sigma \leftarrow$ LocalSearch($\sigma$) (optional)
8:          if $\sigma < best$ then
9:              $best \leftarrow \sigma$
10:         end if
11:         $\theta_{iter} \leftarrow \theta_{iter} \cup \sigma$
12:     end for
13:     UpdatePheromone($T, \theta_{iter}, best$)
14: end while
15: return $best$
```

ACO has been used to solve several combinatorial optimization problems, reaching in many cases state of art performances, as shown in [13]. As far as we know, there are no applications of ACO to automated (general purpose) planning, with the exception of very specialized fields like for instance *path planning* [15].

3 Automated planning

Automated planning is a well known AI problem deeply studied in the last years. Several planning models have been proposed for different classes of problems.

In the standard reference *classical planning model* or *STRIPS model* [24], world states are described in terms of a finite set $\mathcal{F}$ of propositional variables under Closed World Assumption (CWA), i.e., a state s is represented with a subset of $\mathcal{F}$, containing all the variables which are true in s.

A Planning Problem is defined by a triple $(\mathcal{I}, \mathcal{G}, \mathcal{A})$, in which $\mathcal{I} \subseteq \mathcal{F}$ is the initial state, $\mathcal{G} \subseteq \mathcal{F}$ denotes the goal, and $\mathcal{A}$ is a finite set of actions.

An action $a \in \mathcal{A}$ is described by a triple $(pre(a), add(a), del(a))$, where

(i) $pre(a) \subseteq \mathcal{F}$ is the set of preconditions: a is executable in a state s if and only if $pre(a) \subseteq s$

(ii) $add(a), del(a) \subseteq \mathcal{F}$ are, respectively, the sets of positive and negative effects of the action a. If an action a is executable in a state s, the state resulting after its execution is

$$Res(s, a) = (s \setminus del(a)) \cup add(a) \tag{3}$$

otherwise $Res(s, a)$ is undefined.

A linear sequence of actions $(a_1, a_2, \ldots, a_n)$ is a plan for a problem $(\mathcal{I}, \mathcal{G}, \mathcal{A})$ if a_1 is executable in $\mathcal{I}$, a_2 is executable in $s_1 = Res(\mathcal{I}, a_1)$, a_3 is executable in $s_2 = Res(s_1, a_2)$, and so on. A plan $\pi = (a_1, a_2, \ldots, a_n)$ is a solution plan for $(\mathcal{I}, \mathcal{G}, \mathcal{A})$ if $\mathcal{G} \subseteq s_n$.

Usually a planning problem is stated as a pure search problem, in which the purpose is to find, if any, a solution plan. In this setting, the computational problem of determining the existence of a solution to a planning problem is known to be PSPACE–complete [9].

Among the many algorithmic solutions for planners it is worth to point out three main approaches. Designing *planning ad–hoc algorithms* has been the first technique, where Partial Order Planning has dominated until mid–1990's. More recently one of the most successful ad–hoc resolutive method is the graph–based approach introduced in Graphplan [6], also used in FF [19] and LPG [16]. A second approach consists in *planning problem mapping*, i.e., to translate a planning problem into a different combinatorial problem, like logical formula satisfiability [1, 10, 20], integer programming [23] or constraint satisfaction problems, and to use fast solvers designed for these problems. A third approach, which is sometimes combined with ad–hoc methods, is to formulate *planning as heuristic search problem*, as done in HSP [8], FF [19], Fast Downwards [17] and Lama [25].

The classical planning model has been extended in many directions in order to increase expressivity and reasoning capabilities. Among the many extensions, it is worth to mention the introduction of numerical fluents in planning problems: in these models states and goals are also described in terms of numerical variables, actions can also require numerical preconditions and can change the value of numerical variables as effect. In other extensions, like *probabilistic planning*, states are described by probability distributions, actions can have stochastic effects and plans can guarantee to reach the goal only with a certain probability. In *planning with preferences* some goals and preconditions can be neglected without invalidating the action applicability or the problem solvability, but only reducing the plan quality, (notion of "soft" preconditions and goals).

A planning problem can also be formulated as the optimization problem to find, if any, a solution plan which minimizes (or maximizes) a given metric function f. The most common metric functions used in the different planning models are:

- in propositional planning, the plan length;
- in non uniform action cost planning, the sum of the action costs;
- in numerical planning, any expression of the numerical fluents (usually a rational function);
- in planning with preferences, a function which takes into account the violation costs of soft preconditions and goal utilities;
- in probabilistic planning, the probability of reaching the goals or the expected utilities of the reached goals.

The optimization version of the planning problem has been empirically shown to be much harder than its pure search counterpart. Indeed, even for the basic metrics in propositional planning models, only a few planners are really able to find optimal solutions for non trivial problem instances, as shown in the results of the last planning competition IPC-6 [18]. However a larger number of very efficient suboptimal planners have been also proposed especially to cope with large problem instances and more complex metrics and planning models.

4 Planning with ACO

The main idea of this work is to use an ACO approach to solve optimization planning problems. The approach is based on the strong similarity between the process used by ants to build solutions and the way used by state—based planners to find solution plans. Therefore, we have defined an ACO algorithm which handles a colony of planning ants with the purpose of solving planning problems by optimizing solution plans with respect to the overall plan cost.

According to the main features of ACO the ants–planners of the colony are stochastic and heuristic–based. In particular, the importance of an informed heuristic is threefold:

(i) to find solution plans, the heuristics primarily help the ants to find solutions for planning which is a computationally hard problem;
(ii) to drive ants choices towards good (maybe optimal) solution plans;
(iii) to compare plans, the heuristic function is used to compare plans found by different ants.

Each ant–planner executes a forward search, starting from the initial state $\mathcal{I}$ and trying to reach a state in which the goal $\mathcal{G}$ is satisfied. The solution is built step by step by adding solution components. The choice of the components structure, i.e., pheromone model, is one of the main points to address in ACO systems. In general each component is a piece of the problem solution so, in the planning context, the most intuitive and simple choice fell on single actions as possible components. However, this solution is too little informative because it should contain information about the context in which the action will be applied. Four different pheromone models are proposed and described in Section 4.2.

At each step, the ant–planner search process performs a randomized weighted selection of a solution component c which takes into account both the pheromone value $\tau(c)$ associated to the component and the heuristic value $\eta(a)$ computed for each action a executable in the current state and related to the chosen solution component. Once an action a has been selected, the current state is updated by means of the effects of a.

The construction phase stops when at least one of the following termination condition is verified

1. a solution plan is found, i.e., a state where all the goals are true is reached;
2. a dead end is met, i.e., no action is executable in the current state;
3. an upper bound $L_{\max}$ for the number of execution steps is reached.

An alternative way to forward search, which has been considered, is to use a colony of backward ant–planners which start the search phase from the goal and try to build solution plan by *regression*. In this case an appropriate heuristic function should be designed accordingly to be used in a backward search. The backward approach have been experimented in [3], but the first empirical results, not reported here, show that the forward method seems to outperform the backward version, although the issue will need further investigations.

In the ACOPlan algorithm here described, all the ant–planners in the colony are homogeneous and they can fully share information about the pheromone values distribution. In general, as we will discuss in the conclusions, it is possible to define colonies composed by heterogeneous planners with different degrees and types of information sharing.

4.1 The algorithm

The generic ACOPlan algorithm [2–5] is described in Fig. 2. Let $(\mathcal{I}, \mathcal{G}, \mathcal{A})$ be the planning problem, the optimization process is iterated for a given number N of iterations, in which a colony of n_a ants build plans with a maximum number of steps $L_{\max}$. At each step, each ant chooses an action among the executable ones by the *ChooseAction* function that encodes the transition probability function previously described. When all ants have completed the search phase, the best plan π_{iter} of the current iteration is selected and the global best plan π_{best} is possibly updated. Finally, the pheromone values of the solution components are updated by means of the function *UpdatePheromone* that implements the updating rules presented in Section 4.4. Relevant parameters are c_0 which denotes the initial value for the pheromone, ρ which represents the evaporation rate and σ that is a parameter of the pheromone update rule (see (6)).

Algorithm 2 The algorithm ACOPlan

```
1: $\pi_{best} \leftarrow \emptyset$
2: $InitPheromone(c_0)$
3: for $g \leftarrow 1$ to $N$ do
4:     for $m \leftarrow 1$ to $n_a$ do
5:         $\pi_m \leftarrow \emptyset$
6:         $s \leftarrow \mathcal{I}$
7:         $A_1 \leftarrow$ executable actions in $\mathcal{I}$
8:         for $i \leftarrow 1$ to $L_{max}$ while $A_i \neq \emptyset$ and $\mathcal{G} \nsubseteq s$ do
9:             $a \leftarrow ChooseAction(A_i)$
10:            extend $\pi_m$ with $a$
11:            $s \leftarrow Res(s, a)$
12:            $A_{i+1} \leftarrow$ executable actions on $s$
13:        end for
14:    end for
15:    find $\pi_{iter}$
16:    update $\pi_{best}$
17:    $UpdatePheromone(\pi_{best}, \pi_{iter}, \rho, \sigma)$
18: end for
```

4.2 The pheromone models

The simulation of the stigmergic mechanism typical of natural ants is crucial for the optimization process, then the choice of the structures where pheromone is deposited deserves a deep investigation.

An effective pheromone model should be simple to compute but enough informative to guide the search. This consideration leads to exclude some very intuitive but too simple pheromone models, like the one that consider only single actions

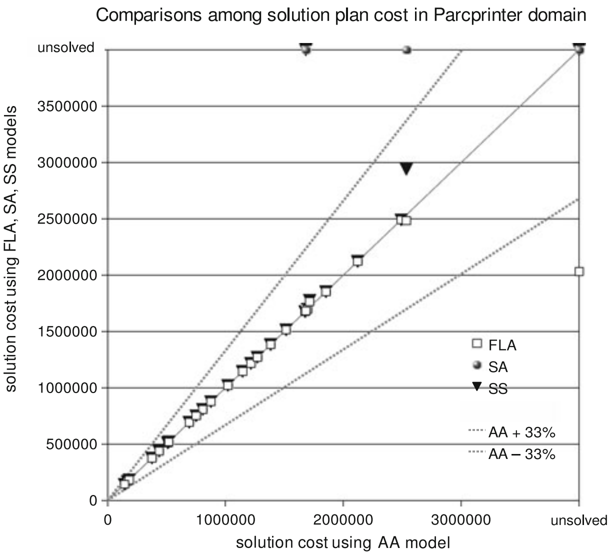

Fig. 2 Scatter plot comparing the solutions of the AA model versus the other models in terms of solution cost in the *Parcprinter* domain

as possible components. This model is indeed too little informative: for instance it cannot distinguish between situations in which executing an action a is useful from those ones in which other actions appear to be better than a.

A useful solution component is then a structure characterizing the context in which an ant–planner can choose a specific action, easy to represent and to compute. A good component for action choice should be enough informative to distinguish the context of most successful choices from the worst ones. On the other hand the characterization of the component should not be too much detailed in order to allow the pheromone to deposit in a significant quantity.

The following four pheromone models has been considered and experimented with the ACOPlan algorithm.

State–State (SS) model A component is defined by the current state s. This is one of the most expensive pheromone model from the space complexity point of view, because the number of possible states is exponential with respect to the problem size. Anyway the storage of the pheromone values can exploit the existence of the $s$tate–cache data structure, as described in Section 4.3.

State–Action (SA) model The pheromone value τ depends on the current state s and on the action a to be executed. This model is even more expensive than SS, because for each state s there can exist several actions executable (and chosen) in s. On the other hand, the pheromone values can be interpreted in terms of a *preference policy*: $\tau(a, s)$ represents how much it is desirable, or it has been useful, to execute a in state s.

Action–Action (AA) model In this model a notion of *local history* is introduced: the pheromone depends both on the action a under evaluation and on the last executed action a', i.e., the pheromone is a function $\tau(a, a')$. Considering only the previous action is the simplest way in which the action choice can be directly influenced by history of previous decisions. AA allows a manageable representation and defines a sort of local first order Markov property.

Fuzzy level–Action (FLA) model The basic idea underlying this model is to associate the action under evaluation with the plan time step, i.e., the plangraph level, where it is executed. Since the limited number of levels, such approach has a more tractable space complexity with respect to the *State–Action* model. On the other hand a pure *Level–Action* model would present the drawback that the pheromone of an action at a time step t cannot be used in other close time steps, while it is often likely that an action desirable at certain time step, say 2, will also be desirable at close time steps, say 1 and 3. To solve this problem the *Fuzzy level–Action* model which fuzzifies the *Level–Action* representation just described has been introduced: when pheromone is distributed over an action a executed at time step t is also spread in decreasing quantity over the same action in the close time steps, conversely the pheromone level computed for an action a to be executed at time t is computed as the weighted average of the pheromone values $\tau(a, t')$ for a and for $t' = t - W, \ldots, t + W$ computed with the Level–Action model, where weights, i.e., the spread distribution, and the time window W are parameters of the model.

The former pheromone models have been empirically compared by systematic experiments whose results are described in the next section. The experiments have allowed to draw general principles and guidelines to develop effective pheromone models which are discussed and motivated in the Section 6.

4.3 The heuristic *FFAC* for actions costs

The heuristic function is a key feature of ACOPlan because it directly affects the transition probability function (1) used to synthesized the solution plan. The heuristic value for a component c is defined by

$$\eta(c) = \frac{1}{h(s_c)}$$

where h is an heuristic function which evaluates the state s_c resulting from the execution of the action a_c associated to the component c in the current state.

The first version of ACOPlan [2–5], which solves problems in the classical planning model, uses the heuristic *FF* introduced by Hoffmann and Nebel in [19] because it provides a good estimation of the distance between a generic state and the goal state in terms of the number of the actions, i.e., plan length.

In the case of planning with action costs, this heuristic is no longer informative because it would neglect the action costs.

At this aim, in this new version of ACOPlan we use a heuristic function which takes into account of the action costs and which is very similar to the heuristic function used in SAPA [12] with the sum propagation for action cost aggregation. We obviously use, as done in classical planning, a constant duration for each action and usual timing for preconditions and effects (respectively at the beginning and at the end of the action). In order to keep this paper self–contained we report here a detailed description of the method used to compute the heuristic function.

This heuristic exploits the idea of FF to use the *relaxed planning graph* structure to compute a relaxed solution plan and use the provided information to estimate the cost of the actual solution. The relaxed planning graph is derived by the classical planning graph introduced in [6] by ignoring the delete list, i.e., the negative effects of the actions. A relaxed planning graph is then a leveled graph with two kinds of nodes: facts and actions. A fact level F_k is the set of the facts that are true at the step k, while the action level A_k is built using the actions that are executable in the states represented by the fact level F_k. Given a fact level F_k and the corresponding action level A_k, the successor fact level F_{k+1} is computed using only the positive effects of the actions in A_k. The graph begins with the initial fact level $F_0 = \mathcal{I}$.

To compute the heuristic value of a state s, first a relaxed planning graph from s to $\mathcal{G}$ is built, then a relaxed plan π^+ is extracted from it by a backward search. The technique used for the π^+ extraction is different to the one used in the FF system because the concept of *action difficulty* has been redefined taking into account the action costs. The cost of the relaxed plan $c(\pi^+)$ is the heuristic value that estimates the cost needed to reach the goals by the current state s:

$$c(\pi^+) = \sum_{a \in \pi^+} c(a)$$

During the graph construction phase a minimum estimated cost c_k is assigned to each fact and each action for every level k of the graph in the following way:

for each $a \in A_k$

$$c_k(a) = c(a) + \sum_{f \in pre(a)} c_k(f). \tag{4}$$

for each $f \in F_k$

$$c_k(f) = \min \left\{ c(a) + \sum_{b \in pre(a)} c_{k-1}(b) \ : \ a \in A_{k-1},\ f \in add(a) \right\}. \tag{5}$$

while, for each $f \in F_0$, we have $c_0(f) = 0$.

Then, new levels are added to the graph until no changes arise, even if goals have been already reached. In other words, the construction process stops when all the goals are present in the graph, no new facts or actions are added and the costs of facts (actions) at the last level L (or $L - 1$ for the actions) are the same they have at the previous level $L - 1$ (or $L - 2$ for the actions).

It is worth to note that the preliminary costs estimation introduced by the formulas (4) and (5) depends on the level in the relaxed planning graph, i.e., we do not have static costs as in the heuristic proposed by Keyder and Geffner in [22] but dynamic costs depending on the level at hand.

The entire process of computation of the heuristic function is shown in Algorithm 3 where the core technique used to compute the relaxed plan π^+ relies in functions λ and *BestAction*. The function λ, applied to an action a or to a fact f, returns the first level k in which $c_k(a)$ or $c_k(f)$ gets the minimum value, while the function *BestAction(g,k)* returns the action $a \in A_{k-1}$ with the minimum *difficulty* which can adds g at the level k. The concept of action difficulty is defined in this context by the cost of the action, i.e., $difficulty_{k-1}(a) = c_{k-1}(a)$.

Algorithm 3 Computing FFAC

```
1:  function FFAC($s$)
2:      build the relaxed planning graph from $s$ and compute $c_k$'s
3:      for $k \leftarrow 0$ to $L$ do $G_k \leftarrow \{g \in \mathcal{G} : \lambda(g) = k\}$
4:      $\pi^+ \leftarrow \emptyset$
5:      for $k \leftarrow L$ downto 0 do
6:          while $G_k \neq \emptyset$ do
7:              take $g$ from $G_k$
8:              $a \leftarrow BestAction(g, k)$
9:              add $a$ to $\pi^+$
10:             for each $p \in pre(a)$ do add $p$ to $G_{\lambda(p)}$
11:             for each $e \in add(a)$ do remove $e$ from $G_k$
12:         end while
13:     end for
14:     return $c(\pi^+)$
15: end function
```

Note that when an action a is added to the plan π^+, the effects of a are deleted from G_k. This feature allows to take into account the so called *positive interactions*, i.e., it avoids to use a new action a' to reach a goal g, when an already selected another action also reaches g. This feature is one of the most important inheritance we receive by FF: it allows to the heuristic FFAC to differ in one more point to the heuristic h_{sa} used in *FF*(h_a) [22] that cannot take into account *positive interactions*. Hence, there exist problems for which h_{sa} overestimates the number of actions, and possibly the cost too, needed to reach the goals from a given state with respect to the estimate computed by *FFAC*.

As *FF*, also *FFAC* can compute the set of *Helpful Actions*, which are the actions in the first step of π that add some subgoals present in the level F_1 during the relaxed plan extraction process. *Helpful Actions* can be considered as *privileged* actions, i.e., actions whose execution is more desirable than other ones. Therefore, in ACOPlan the heuristic value of states reached by the execution of *Helpful Actions* can be decreased, so that they have a greater probability of being chosen.

Since the computation of the heuristic function is computationally demanding and the number of evalutation for h is huge (each ant at each step should compute h for each reachable state), a *s*tate–cache data structure is used to store heuristic values (and other informations, like the pheromone value for the SS model) for each visited state. This structure is implemented through a hash table to speed up the search process. Moreover in order to avoid memory problems the size of the hash table is bounded and table entries are eventually removed using a Least Recently Used strategy.

4.4 Plan comparison and pheromone updating

A critical point of the optimization process is the ability of comparing plans found by the colony of planner ants. This is particularly important in pheromone updating phase where the best plan of the iteration must be selected, as well as in the general ACOPlan which returns the best plan found in all the iterations.

Any comparison criteria should obviously prefer a *solution plan* to a *non solution plan*. On the other hand comparison of two *solution plans* π, π' can be easily based on actions costs, because it is possible to compute their respective costs and prefer the plan with the lowest cost or, when they are equal in cost, let prevail the one with the lesser plan length metric, either sequential or parallel.

A comparison criteria cannot be easily defined when both plans π and π' are not solution plans. In this case using only the overall action cost of the non solution plan makes no much sense, since a cheap plan can be far away from any solution. In other words, it would be useful to take into account both the distance from the goals and cost of a non solution plan.

In order to evaluate a plan π, it is possible to combine the value of the heuristic function *FFAC* on the best state s ever reached by π with the cost of reaching s.

More formally, let $\pi = (a_0, a_1, \ldots, a_n)$ be a plan and let $s_{i+1} = Res(s_i, a_i)$, for $i = 0, \ldots, n$, with $s_0 = \mathcal{I}$, be the states reached by π, we define the penalty of plan π as

$$p(\pi) = \min_{k=0,\ldots,n} \left(\sum_{i=0}^{k} c(a_i) + \omega FFAC(s_k) \right)$$

where ω is a parameter which is used to enhance the influence of the second factor, the distance from s_k to the goal state, with respect to the first one, the cost to reach the state s_k. The parameter ω is usually set to a high value (e.g. 10), because the distance to the goal state has a greater importance. Then π is preferable to π' if $p(\pi) < p(\pi')$.

The *pheromone update phase* evaporates all the pheromone values and increases the pheromone value of the components belonging to π_{iter} and π_{best} according to the formula

$$\tau(c) \leftarrow (1 - \rho) \cdot \tau(c) + \rho \cdot \Delta(c) \tag{6}$$

where

$$\Delta(c) = \begin{cases} \sigma & \text{if } c \text{ belongs to } \pi_{\text{iter}} \\ 1-\sigma & \text{if } c \text{ belongs to } \pi_{\text{best}} \\ 1 & \text{if } c \text{ belongs to both} \\ 0 & \text{otherwise} \end{cases}$$

and σ is usually set to $\frac{2}{3}$ which increases a component belonging to the best plan in the iteration by a double quantity with respect to a component belonging to the best plan so far, thus enforcing the exploration, instead of the exploitation.

5 Experimental evaluation

This section presents and discusses the results of experiments held with ACOPlan optimizing the overall plan execution costs. The previously introduced pheromone models are compared, and comparisons are also made with state of the art satisficing planners both with respect to the solving capability and with respect to the quality of the synthesized solutions.

The benchmarks domain problems from the recent planning competition IPC-6 [18] has been used in the experiments. The results presented here refers to the domains *Elevators*, *Openstacks*, *Parcprinter*, *Pegsol*, *Transport* and *Woodworking*.

Since the behavior of ACOPlan depends on many parameters, a preliminary phase of parameters tuning has been held by systematic tests in order to establish the ACOPlan general setting which has been used in all four pheromone models. For more information and details about this experiments see [26]. The next phase of actual experiments has then been conducted with the ACOPlan settings: 10 planner–ants, 5,000 iterations, $\alpha = 2$, $\beta = 5$, $\rho = 0.15$, $c_0 = 1$ (initial pheromone value), $k = 0.5$ (decreasing rate of the heuristic value of states reached by the execution of *Helpful Actions*). Moreover, since ACOPlan is a not deterministic planner, the results collected for each single problem instance are the median values obtained over 50 runs. The experiments were run on a Linux machine with a 2.4 GHz quad core processor with 4 GB of RAM.

5.1 Benchmark domains

In this section a brief description of the tested domains is given. Each domain contains 30 problems numbered p01,...,p30 which can be grouped in three ordered sets

$$p01, \ldots, p10$$

$$p11, \ldots, p20$$

$$p21, \ldots, p30$$

which can be roughly considered of increasing difficulty due to the increasing number of objects in the problems.

Elevators A building of N+1 floors that can be separated in blocks of size M+1 is served by elevators. Adjacent blocks have a common floor. For example, suppose

N=12 and M=4, then we have 13 floors in total (ranging from 0 to 12), which form 3 blocks of 5 floors each, being 0 to 4, 4 to 8 and 8 to 12.

The building has K fast (accelerating) elevators (with capacity of X persons) that stop only in floors that are multiple of M/2 and, within each block, there are L (with capacity of Y persons) slow elevators, that stop at every floor of the block.

There are costs associated with each elevator starting/stopping and moving. In particular, fast (accelerating) elevators have negligible cost of starting/stopping but have significant cost while moving. On the other hand, slow (constant speed) elevators have significant cost when starting/stopping and negligible cost while moving.

There are several passengers, for which their current location (i.e., the floor they are) and their destination are given. The planning problem is to find a plan that moves the passengers to their destinations minimizing the cost.

Parcprinter This domain models the operation of the multi-engine printer, for which one prototype is developed at the Palo Alto Research Center (PARC). This type of printer can handle multiple print jobs simultaneously. Multiple sheets, belonging to the same job or different jobs, can be printed simultaneously using multiple Image Marking Engines (IME). Each IME can either be *color* or *mono*. Each sheet needs to go through multiple printer components such as feeder, transporter, IME, inverter, finisher and need to arrive at the finisher in order. Thus, sheet (n+1) needs to be stacked in the same finisher with sheet n of the same job, but needs to arrive at the finisher right after sheet n (no other sheet stacked in between those two consecutive sheets). Given that the IMEs are heterogeneous (mixture of color and mono) and can run at different speeds, optimizing the operation of this printer for a mixture of print jobs, each of them is an arbitrary mixture of color/b&w pages that are either simplex (one–sided print) or duplex (two–sided print) is a hard problem.

The objective function is to minimize the printing cost. For example, using a more expensive color IME to print a black&white page costs more than using a mono IME. However, the cost tradeoff may not be clearcut if the feeder, where the blank sheets originally reside at, is closer to the mono IME than to the color IME.

Openstacks A manufacturer has a number of orders, each of them for a combination of different products, and it can only makes one product at a time.

The total required quantity of each product is made at the same time (because changing from making one product to making another requires a production stop). From the time that the first product included in an order is made to the time that all products included in the order have been made, the order is said to be "open" and during this time it requires a "stack" (a temporary storage space). The problem is to order the making of the different products so that the maximum number of stacks that are in use simultaneously, or equivalently the number of orders that are in simultaneous production, is minimized (because each stack takes up space in the production area). The goal is to minimize the total number of stacks.

The problem is NP-hard.

Pegsol This domain models the well known Peg solitaire (the English version), that is a board game for one player involving movement of pegs on a board with holes. Pegs are arranged on the board such that at least one hole remains. By making draughts–like moves, pegs are gradually removed until no further moves are possible or some goal configuration is achieved. A valid move is to jump a peg orthogonally over an adjacent peg into a hole two positions away and then to remove the jumped peg. The difficult of each problem is defined by the number and the initial position of the pegs in the board. The goal is defined by the position that the last (unique) peg has to reach. The planning domain provides three actions defining a starting jump move, a continuing jump move and an ending move. The action costs are used to define the number of jump sequences as the optimization criterion. The problem is NP–hard.

Transport This domain models a list of packages, a list of vehicles and a list of locations. Packages must be picked up to some vehicle (characterized by a given capacity) in order to be moved and then they must be dropped from the vehicle to reach a destination; vehicles can drive between two different locations. Each drive action has a cost proportional to the distance between the two locations, while the pickup and drop actions have unitary cost. The objective is to deliver some given packages into given locations starting from different initial locations (both for packages and vehicles). The objective is to minimize the total cost (i.e., minimize the covered distance).

Woodworking This domain models the operations in a woodworking where wood objects are worked. Objects are worked in parts that can be modeled by a grinder or a planer in order to obtain different treatment of the surface: smooth or very-smooth; they can be cut by two different types of saw (normal or highspeed) in order to obtain border with different size: small, medium or large; they can be colored by two types of varnisher (immersion-varnisher and spray-varnisher) or by a glazer. All these operations are modeled by specific actions with different costs that, in general, require the part under consideration to be previously worked by another machine (for instance, the varnishing operations require the object to be smooth). The goal is to obtain object parts with given manufacturing minimizing the total-cost.

5.2 Comparing pheromone models

This group of experiments aims at comparing and evaluating the ACOPlan pheromone models introduced in Section 4.2. Initially the models have been evaluated with respect to their problem solving ability, i.e., to find which one can solve the greatest number of problems. In Table 1 it is shown, for each domain, the percentage of problem instances (runs) solved in the upper bound time of 900 s.

In order to understand if the differences from the solving capability point of view observed in the experiments are statistically significants, we have applied the χ^2 test to compare the probabilities of success of the different pheromone models. In this test we have considered the number of the solved instances with respect the total

Table 1 Percentages of problems solved in 900 s for each IPC–6 domain

	AA (%)	FLA (%)	SA (%)	SS (%)
Elevators	87	83	81	82
Openstacks	100	100	100	100
Parcprinter	80	80	73	74
Pegsol	96	96	96	96
Transport	82	84	82	83
Woodworking	100	100	100	100

number of runs; we have tested 6 domains containing 30 problems each, and for each problem we run the planner for 50 times. Therefore we have submitted to each pheromone model 9000 problem instances.

The number of successes for each model is shown in brackets in the heading row/column in Table 2 that shows the results of such a statistic. Each cell in the table contains the χ^2 value and the p-value obtained comparing the model in the row heading and the model in the column heading; Low p-values allow to reject the null hypothesis (the two model performance are equal) in favour of the alternative hypothesis to have better performance for the first model.

A first point which can be noticed is that *AA* and *FLA* have comparable performance with a little difference in favour of *AA* while, setting the confidence level at 99.5%, *SS* and *SA* are statistically worse than them with a relevant gap.

On the other hand a very encouraging new result has been obtained: ACOPlan with *FLA* pheromone model performs very close to the *AA* model, although, as noted in the previous section, *FLA* is much more manageable with respect to space complexity.

Moreover we have compared the pheromone models from the solution quality point of view. In this case we have computed and compared for each solved problem the median values for solution cost over the 50 runs; these values have been plotted in the graphs in Figs. 1, 2, 3 and 4.

These plots give a graphical representation of the performance of the AA model compared with the other models in terms of solution cost. The figures show respectively the results for the domains *Elevators*, *Parcprinter*, *Transport* and *Woodworking*. Each symbol represents a problem instance; if the symbol is under the solid diagonal line, then the corresponding model has a median value for solution cost that is lower than the median value of the AA model; otherwise (i.e., the symbol is over the solid line), the opposite is true (i.e., the compared model is better than the AA model). The plots have two additional dotted lines delimiting the area where solutions with a cost differing of less than 33% from the AA solution fall. The plots for the *Openstacks* and *Pegsol* domains are not significant and have not been presented; this is due to the excellent results we have obtained in these domains where all the models find in most the cases the same (optimal) solution as can be noted in the raw data presented in Tables 8 and 10.

A first comment on the overall plan quality result is that the prevalence of the AA and FLA models and the bad performance of SA and SS models observed on the number of solved problems (see Tables 1 and 2) is generally confirmed for the quality found. In other words the problem solving ability and the optimization ability of all the pheromone models seems to follow a similar pattern.

Table 2 Statistical analysis of the pheromone model comparisons by means of χ^2 test

	FLA (8155)	SS (8021)	SA (7990)
AA (8194)	$\chi^2 = 1.01$ $p = 0.16$	$\chi^2 = 18.61$ $p = 8.01 \cdot 10^{-6}$	$\chi^2 = 25.49$ $p = 2.23 \cdot 10^{-7}$
FLA (8155)	–	$\chi^2 = 10.95$ $p = 4.67 \cdot 10^{-4}$	$\chi^2 = 16.36$ $p = 2.61 \cdot 10^{-5}$
SS (8021)	–	–	$\chi^2 = 0.54$ $p = 0.23$

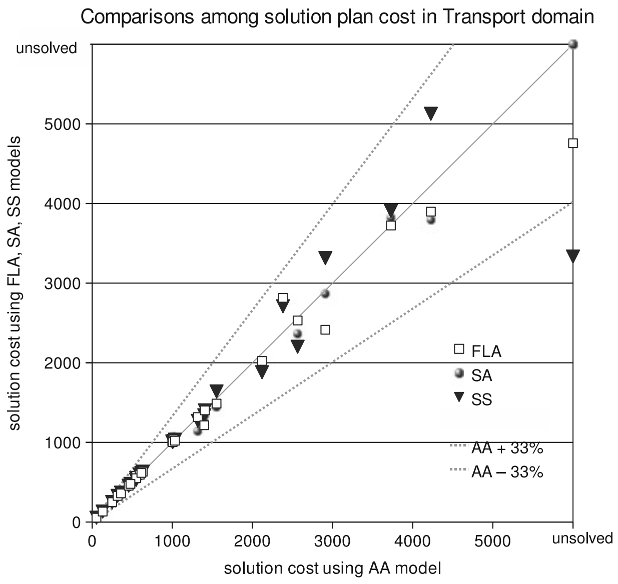

Fig. 3 Scatter plot comparing the solutions of the AA model versus the other models in terms of solution cost in the *Transport* domain

Domains which are considered "difficult", such as *Elevators*, *Transport* and *Woodworking*, show a prevailing bad performance for the component models SS and SA. The intuition would suggest instead that SS and SA characterize the context where a choice is operated by ants, since a pair of states represents the most detailed description of a local context, and the best performance of more simplified contexts such as AA pair and FLA seems to be counterintuitive. A careful analysis has been

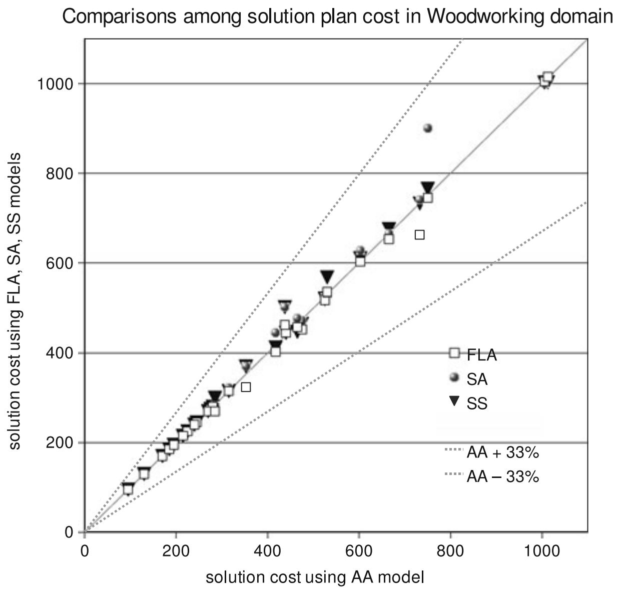

Fig. 4 Scatter plot comparing the solutions of the AA model versus the other models in terms of solution cost in the *Woodworking* domain

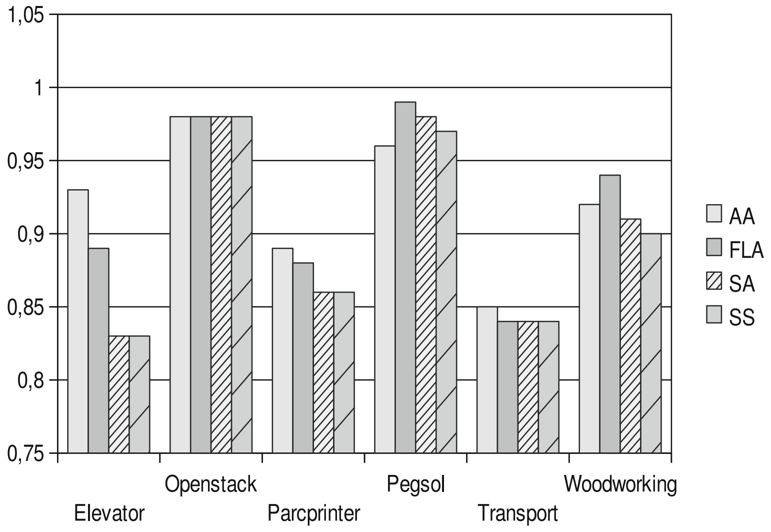

Fig. 5 For each domain, each column represents a synthesis of the average quality found by means of each model. AA(Action-Action), FLA(Fuzzy-Level-Action), SA(State-Action), SS(State-State)

done of pheromone distribution for SS and SA on the worst performing problem instances, such as *elevators09*, *parcprinter19*, *transport15*. The analysis has revealed that the difficult seems to be caused by a too high level of details, i.e., in the proliferation of components. In other words if each different state originates one (or more) different component, similar successful states, which could differ for just one fluent value, do not accumulate enough pheromone, a fact which prevent them to be preferred in the ants choices.

We have observed that in situations where SS and SA are performing bad, the pheromone is spread over too many components, which are not likely to appear again in different runs of the ants, even if the plans found in the run slightly differ from the previous ones. On the other hand a less detailed description of the context, like in the AA model, allows the accumulation of pheromone over components which appear more easily in different successful and similar runs.

Let us consider, for example, two successful plans $\pi_1 = \{A_0, \ldots, A_i, A_{i+1}, \ldots, A_n\}$ and $\pi_2 = \{A_0, \ldots, A_{i+1}, A_i, \ldots, A_n\}$ which differs only for two consecutive switched

Table 3 Results for the Wilcoxon test for the pheromone model comparisons

	AA	SA	SS
FLA	$z = -1.71$ $p = 0.0043$	$z = -3.66$ $p = 0.0001$	$z = -3.02$ $p = 0.0013$
AA	–	$z = -1.91$ $p = 0.0028$	$z = -2.56$ $p = 0.0005$
SA	–	–	$z = 0.78$ $p = 0.0218$

Each cell contains the z-value and p-value obtained comparing the model in the row heading versus the model in the column heading

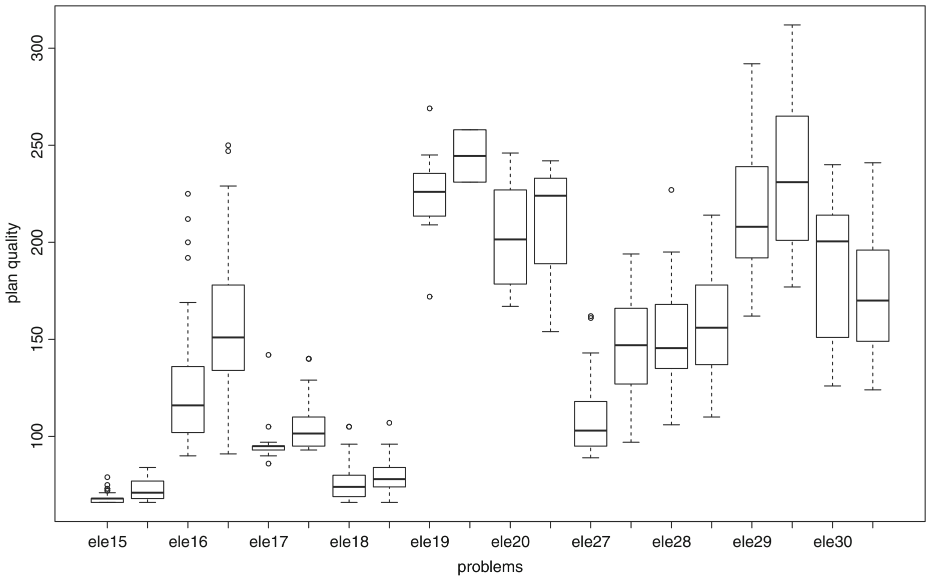

Fig. 6 Distribution of quality of solution plans found in 50 runs for **Elevators** domain with AA and FLA pheromone model. For each problem, the **first box** represents the distribution of quality values obtained by means of **AA** pheromone model, while the **second box** represents the distribution of quality values obtained by means of **FLA** pheromone model

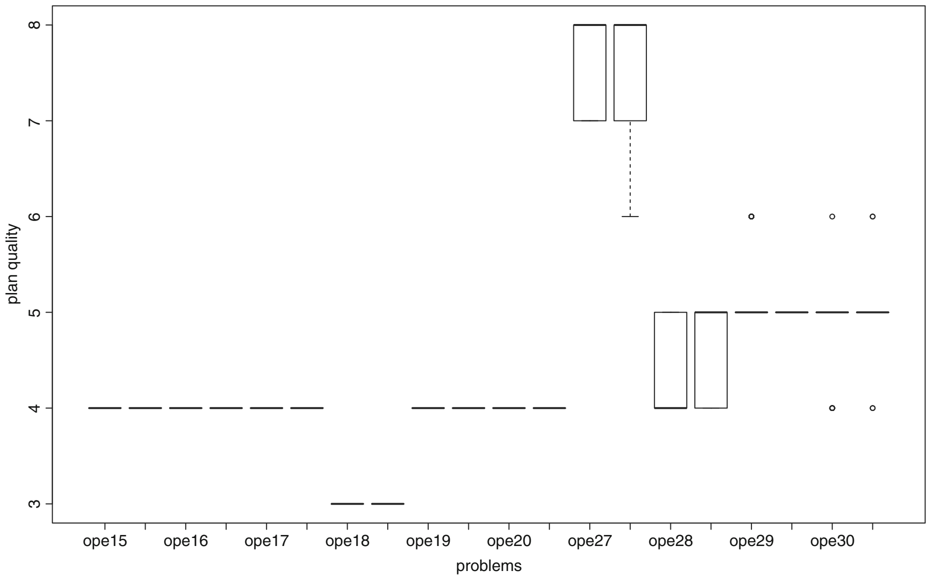

Fig. 7 Distribution of quality of solution plans found in 50 runs for **Openstacks** domain with AA and FLA pheromone model. For each problem, the **first box** represents the distribution of quality values obtained by means of **AA** pheromone model, while the **second box** represents the distribution of quality values obtained by means of **FLA** pheromone model

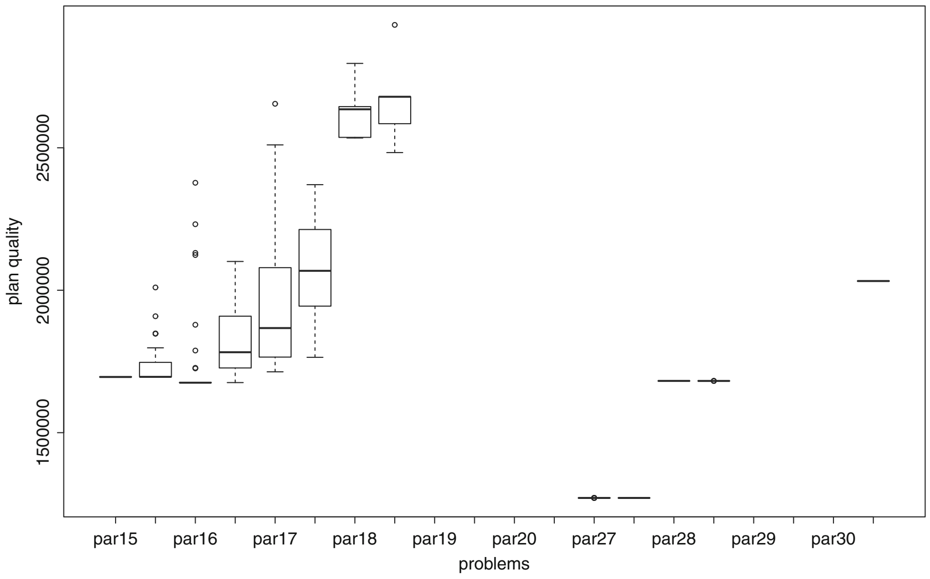

Fig. 8 Distribution of quality of solution plans found in 50 runs for **Parcprinter** domain with AA and FLA pheromone model. For each problem, the **first box** represents the distribution of quality values obtained by means of **AA** pheromone model, while the **second box** represents the distribution of quality values obtained by means of **FLA** pheromone model

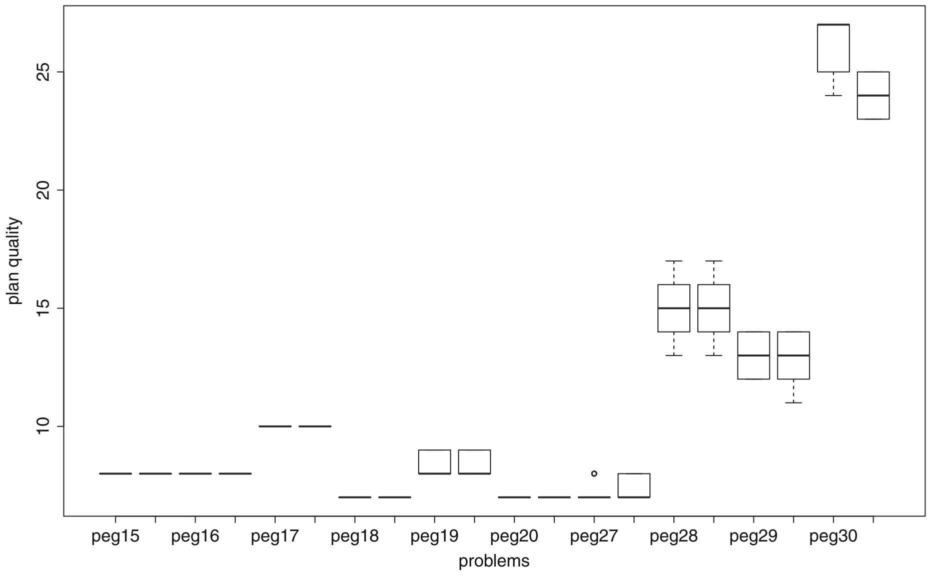

Fig. 9 Distribution of quality of solution plans found in 50 runs for **Pegsol** domain with AA and FLA pheromone model. For each problem, the **first box** represents the distribution of quality values obtained by means of **AA** pheromone model, while the **second box** represents the distribution of quality values obtained by means of **FLA** pheromone model

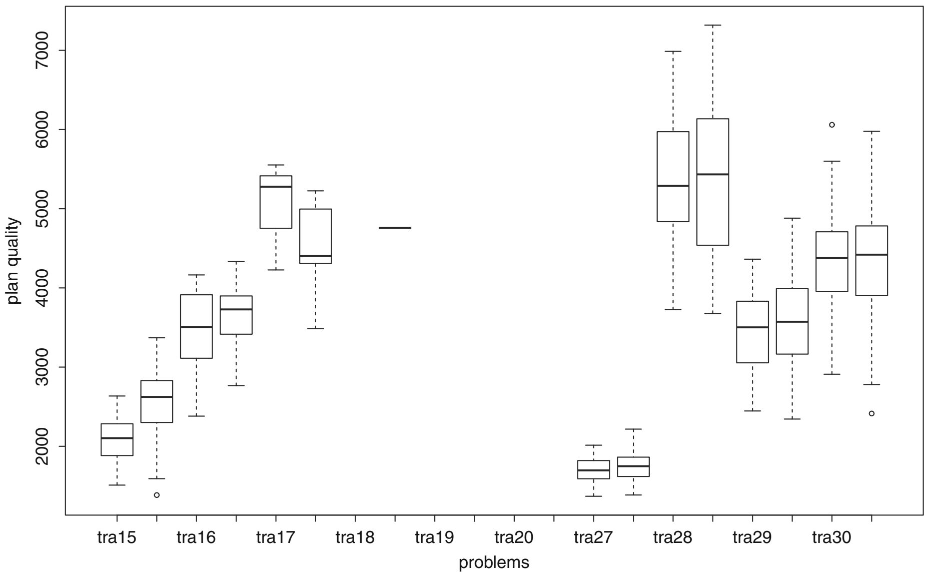

Fig. 10 Distribution of quality of solution plans found in 50 runs for **Transport** domain with AA and FLA pheromone model. For each problem, the **first box** represents the distribution of quality values obtained by means of **AA** pheromone model, while the **second box** represents the distribution of quality values obtained by means of **FLA** pheromone model

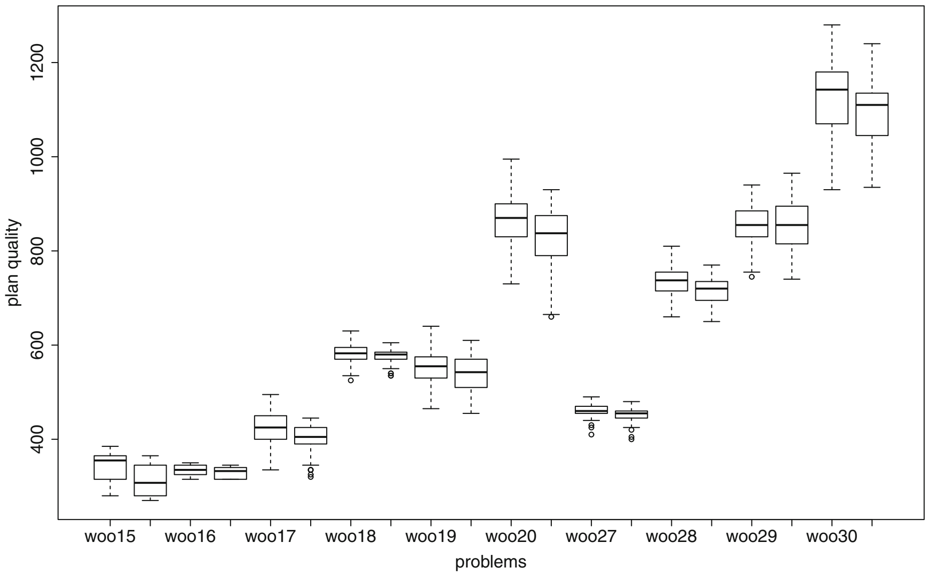

Fig. 11 Distribution of quality of solution plans found in 50 runs for **Woodworking** domain with AA and FLA pheromone model. For each problem, the **first box** represents the distribution of quality values obtained by means of **AA** pheromone model, while the **second box** represents the distribution of quality values obtained by means of **FLA** pheromone model

Table 4 Summary showing the number of solved problems S and the number of problem for which the found solution is the best solution known so far

Domain	FF(h_a)		Lama		LPG		ACO(FLA)		Gamer	
	Solved	BQ	Solved	BQ	Solved	BQ	Solved	BQ	Solved	BQ
Elev	30	2	30	23	30	4	30	18	22	22
Open	30	2	30	30	30	0	30	30	19	19
Parc	16	16	23	12	30	28	27	22	0	0
Peg	29	6	30	25	26	0	30	27	22	22
Trans	29	6	30	22	29	17	28	11	11	11
Wood	30	0	30	9	30	25	30	16	13	13
Total	164	32	173	121	175	74	175	124	87	87

The same 180 problems have been submitted to all the five planners

actions A_i and A_{i+1}. The two plans in the *Action-Action* model will only differ *locally* for the three components (A_{i-1}, A_i) , (A_i, A_{i+1}) and (A_i, A_{i+1}), while in the SS (or in the SA) models they will potentially differ on all the components starting from the state produced by action A_{i-1} onward, till the end of the plans. It is apparent that the earlier a different state is produced the greatest is the impact on the component differences among plans which are similar.

Considering the *global plan quality* shown in Fig. 5, the FLA model still confirms its proximity to the AA model, i.e., FLA is always performing second and it is the best performing one also in the only domain (see *Pegsol* domain) where AA is not. The global quality results for *Openstacks* also point out that it is in general an easy problem domain not sufficiently significant to address quality performance comparisons.

The performance of the FLA model can be explained by reasons similar to the ones supporting the better behavior of the AA model with respect to *state based* components. In fact, using FLA, two similar plans, in the sense of the previous π_1 and π_2, receive pheromone in similar components: the model is *robust* with respect to local differences such as consecutive action switching or action insertion in a plan.

Moreover the FLA model is more flexible with respect to AA. In AA the pheromone for plan π_1 is given, for instance, to (A_i, A_{i+1}) whereas pheromone for plan π_2 is put on component (A_{i+1}, A_i). In FLA instead, switching actions A_i and A_{i+1} will not prevent the two components (A_i, i) and $(A_i, i+1)$ to receive pheromone from both plans, although they will receive a fuzzy different amount. The FLA model is then a worthwhile compromise to obtain a good performance close to the AA without the higher implemental associated cost.

In general FLA tends to learn in which time step (or nearby) to apply an action is useful, while AA tends to learn which action is useful to apply after a given one.

Table 5 Statistical analysis of the data presented in Table 4 by means of χ^2 test

	Lama	LPG	FF(h_a)
ACO	$\chi^2 = 0.03$	$\chi^2 = 29.07$	$\chi^2 = 89.85$
(FLA)	$p = 0.43$	$p = 3.48 \cdot 10^{-8}$	$p < 2.2 \cdot 10^{-16}$
Lama	–	$\chi^2 = 27.01$	$\chi^2 = 86.36$
		$p = 1.01 \cdot 10^{-7}$	$p < 2.2 \cdot 10^{-16}$
LPG	–	–	$\chi^2 = 20.43$
			$p = 3.09 \cdot 10^{-6}$

Table 6 Statistical analysis with the Wilcoxon Signed-Rank test of the data reported in Tables 7–12

	Lama	LPG	FF(h_a)
ACO (FLA)	$z = 1.48$ $p = 0.08$	$z = -3.75$ $p = 0.0001$	$z = -9.54$ $p < 0.0001$
Lama	–	$z = -4.79$ $p < 0.0001$	$z = -10.01$ $p < 0.0001$
LPG	–	–	$z = -7.55$ $p < 0.0001$

In order to understand if the differences between the performance obtained by the tested pheromone models are significative, we carried out a statistical analysis by means of the Wilcoxon Signed-Rank test [11]. The results are shown in Table 3, where each pheromone model has been compared with all the other models considering all the problem instances that are solved by at least one of the models. When a planner does not solve a problem instance, we set the corresponding solution cost to the double of the highest cost found by any other model. The models are pairwise compared; the Wilcoxon test ranks, by increasing numbers, the difference

Table 7 Results for plan quality on *Elevators* domain

Problem	FF(h_a)	Lama	LPG min	LPG median	ACO(FLA) min	ACO(FLA) median	Gamer
elev01	58	42	42.8	42.8	42	42	42
elev02	26	26	26.6	26.6	26	26	26
elev03	99	55	56	56	55	55	55
elev04	52	40	41.2	41.2	40	40	40
elev05	125	55	56.4	60	55	55	55
elev06	118	53	54.8	70.6	53	53	53
elev07	155	62	67.8	67.8	62	106	62
elev08	126	53	58.6	66.6	53	53	53
elev09	305	87	88	88.2	89	103.5	78
elev10	171	99	80	82	103	103	–
elev11	109	56	57	57	56	56	56
elev12	92	54	54.8	54.8	54	54	54
elev13	88	59	60	60	59	59	59
elev14	147	63	64.4	65.4	64	64	63
elev15	128	66	67.4	68.2	66	66	66
elev16	190	87	89.2	98.6	91	106	–
elev17	200	81	83	88.6	93	100	78
elev18	116	61	62.8	69.6	66	86.5	61
elev19	228	147	124.6	140.2	231	244.5	–
elev20	182	88	99	99.2	154	154	–
elev21	48	48	49	49	48	48	48
elev22	78	54	55.8	55.8	55	55	54
elev23	122	69	70.4	70.4	69	69	69
elev24	90	61	57.6	57.6	56	56	56
elev25	150	63	64.8	66.2	63	103	63
elev26	71	48	49.2	49.2	48	48	48
elev27	152	82	83.8	84	97	107	–
elev28	226	82	86.6	89.4	110	110.5	–
elev29	240	157	111.8	116.8	177	177	–
elev30	131	89	86.4	86.4	124	136.5	–

between the solution cost found by the two planners, assigning a minus/plus sign to the rank value according to the sign of the corresponding difference. Since the range of the quality values may vary greatly between domains, or even between problems in the same domain, we normalized the differences by dividing the two values we are comparing with the quality value of the better plan. More formally, let q_1 and q_2 the two values for the solution plan quality obtained by the two planners we are comparing, before applying the Wilcoxon test we have replaced those values with $q_1/\min(q_1, q_2)$ and $q_2/\min(q_1, q_2)$ respectively.

Since a better solution has a lower value for solution cost, a preponderance of negative signs among the signed ranks would suggest that the first planner found solutions with lowest cost (i.e., performs better), while a preponderance of positive signs would suggest the opposite. The null hypothesis is that there is no tendency in either direction, hence that the numbers of positive and negative signs will be approximately equal. In that event, we would expect the value of the sum of the signed ranks to be near zero, within the limits of random variability. Each cell of the Table 3 contains the z-value and the p-value resulting from the application of the test

Table 8 Results for plan quality on *Openstacks* domain

Problem	FF(h_a)	Lama	LPG min	LPG median	ACO(FLA) min	ACO(FLA) median	Gamer
open01	3	2	3.5	3.5	2	2	2
open02	3	2	3.8	3.8	2	2	2
open03	2	2	4.1	4.1	2	2	2
open04	3	3	5.4	5.4	3	3	3
open05	9	4	6.7	6.7	4	4	4
open06	5	2	5	6	2	2	2
open07	7	5	8.3	8.3	5	5	5
open08	10	5	8.6	8.6	5	5	5
open09	11	3	6.9	7.9	3	3	3
open10	11	3	7.2	8.2	3	3	3
open11	9	4	8.5	9.5	4	4	4
open12	13	3	9.8	9.8	3	3	3
open13	9	4	10.1	11.1	4	4	4
open14	13	4	11.4	12.4	4	4	4
open15	12	4	11.7	13.7	4	4	4
open16	15	4	13	13	4	4	4
open17	15	4	13.3	15.3	4	4	4
open18	14	3	13.6	14.6	3	3	3
open19	15	4	13.9	15.9	4	4	–
open20	14	4	15.2	15.2	4	4	–
open21	22	3	14.5	15.5	3	3	–
open22	18	4	17.8	18.8	4	4	4
open23	18	4	17.1	20.1	4	4	–
open24	16	5	18.4	18.4	5	5	–
open25	20	4	21.7	22.7	4	4	–
open26	27	5	18	19	5	5	–
open27	20	6	22.3	23.3	6	6	–
open28	19	4	21.6	22.6	4	4	–
open29	24	5	23.9	24.9	5	5	–
open30	32	4	23.2	24.2	4	4	–

using the model in the row heading as the first model and the model in the column heading as the second model. Negative z-value denotes that the model in the row heading performs better with respect the model in the column heading, while the absolute value of the z-value measures the gap between the two models, the higher the z-value the more significant the difference is. The p-value represents the level of significance.

Setting the confidence level at 99.5% we can conclude that, except for the SA–SS comparison, the models statistically differ in their performance.

Finally, we have analyzed the FLA and the AA models more deeply by studying the distribution of the solution costs. For each domain we have plotted plan cost values for both these models and built the corresponding boxplot.

In each boxplot are represented the results of the 10 most interesting problems of the 30 problems in each domain, i.e., the problems p16..p20, p26..p30 for each domain. The results are very different among the domains, while they are quite similar comparing both models in the same domain (Figs. 6, 7, 8, 9, 10 and 11).

Table 9 Results for plan quality on *Parcprinter* domain

Problem	FF(h_a)	Lama	LPG min	LPG median	ACO(FLA) min	ACO(FLA) median	Gamer
par01	169009	169009	169009	169009	169009	169009	–
par02	438047	438047	438047	438047	438047	438047	–
par03	807114	807114	807114	807114	807114	807114	–
par04	876094	876094	876094	876094	876094	876094	–
par05	1145132	1345190	1145132	1145132	1145132	1145132	–
par06	–	1614228	1514199	1514199	1514199	1514199	–
par07	–	1783237	1383121	1383121	1383121	1387122.16	–
par08	–	2152304	1852217	1852217	1852217	1852217	–
par09	–	2421342	2121255	2121255	2121255	2131257.9	–
par10	–	2690380	2490322	2490322	2490322	2502325.48	–
par11	182808	182808	182808	182808	182808	182808	–
par12	510256	510256	510256	510255	510256	510256	–
par13	693064	693064	693064	693064	693064	693064	–
par14	1020512	1020512	1020512	1020512	1020512	1020512	–
par15	1695507	1746041	1695507	1705326	1695507	1695507	–
par16	1675408	1827010	1675408	1675408	1675807	1676006.5	–
par17	1713576	1915712	1713576	1713576	1764509	1764509	–
par18	2330304	2582974	2330304	2330304	2483103	2483103	–
par19	3353256	3851224	3422278	3468829	–	–	–
par20	2754187	–	2775122	2844535	–	–	–
par21	143411	143411	143411	143411	143411	143411	–
par22	–	375821	375821	375821	375821	375821	–
par23	–	519232	519232	519232	519232	519232	–
par24	–	751642	751642	751642	751642	751642	–
par25	–	–	1215839	1215839	1215839	1215839	–
par26	–	–	1216462	1216462	1216462	1216462	–
par27	–	–	1270874	1270874	1270874	1270874	–
par28	–	–	1681282	1681282	1681381	1681381	–
par29	–	–	2377265	2377364	–	–	–
par30	–	–	2021893	2021893	2032289	2032289	–

More in detail, we can note that in the Figs. 7–9, showing respectively the results for *Openstacks*, *Parcprinter* and *Pegsol* domains, we have obtained very dense distributions that in many cases collapse into a line (all the runs found the same value), while in *Transport* (Fig. 10), *Woodworking* (Fig. 11) domains and especially in *Elevators* domain (Fig. 6) the boxes show a higher degree of dispersion and the presence of some outliers. A possible explanation of this behaviour is that the heuristic $FFAC$ is lesser informative in *Transport*, *Woodworking* and *Elevators* domains causing a slower convergence of the algorithm. In this case greater differences can be found among the different solution plans and this leads to a higher dispersion of the synthesized plans in the solution space.

5.3 Comparing ACOPlan with other planners

This second set of experiments has been devoted to compare ACOPlan with other "state of the art" planners. We have chosen to compare ACOPlan with *Fuzzy*

Table 10 Results for plan quality on *Pegsol* domain

Problem	FF(h_a)	Lama	LPG min	LPG median	ACO(FLA) min	ACO(FLA) median	Gamer
peg01	2	2	2.30	2.30	2	2	2
peg02	5	5	5.40	5.40	5	5	5
peg03	4	4	4.50	4.50	4	4	4
peg04	5	4	4.60	4.60	4	4	4
peg05	5	4	4.70	4.70	4	4	4
peg06	4	4	4.80	4.80	4	4	4
peg07	3	3	3.90	3.90	3	3	3
peg08	9	6	7.00	7.00	6	6	6
peg09	5	5	6.00	6.00	5	5	5
peg10	8	6	7.10	7.10	6	6	6
peg11	9	7	8.10	8.10	7	7	7
peg12	9	8	9.20	9.20	8	8	8
peg13	13	9	11.20	11.20	9	9	9
peg14	12	7	–	–	7	7	7
peg15	12	8	9.30	9.30	8	8	8
peg16	10	8	9.30	9.30	8	8	8
peg17	12	10	11.40	11.40	10	10	10
peg18	14	7	12.40	12.40	7	7	7
peg19	11	8	9.40	9.40	8	8	8
peg20	12	7	9.50	11	7	7	7
peg21	14	8	12.50	14	8	8	8
peg22	8	6	7.50	7.5	6	6	6
peg23	16	8	–	–	8	8	–
peg24	14	8	13.70	13.7	8	8	–
peg25	12	8	10.70	12.2	8	8	–
peg26	14	10	13.80	14.3	10	10	–
peg27	14	8	13.10	15.1	7	7	–
peg28	16	15	–	–	13	13	–
peg29	19	13	18.60	18.6	11	11	–
peg30	–	22	–	–	23	24	–

Level Action pheromone model (*ACO(FLA)*) with FF(h_a) [22], Lama [25], Gamer [14] and LPG [16]. Lama was the winner of the last planning competition in the sequential satisficing track, so it is the natural basis for comparisons. The choice of FF(h_a) is twice motivated: it has been the second best planner resulting from the above track competition and it uses a heuristic function quite close to ACOPlan. LPG has been chosen because it is one of the best performing planners of the last years and moreover it uses a non-deterministic approach. Finally, Gamer has been chosen because it was the winner of the last planning competition in the sequential optimization track: it has been mainly used to obtain the optimal solution quality in order to evaluate the solutions found by the other planners.

All the planners have run on the same machine, with the same limit for the CPU time (varying on the basis of the problem difficulty), and using the most suitable parameters values (if tunable) to improve the quality of the plans found. In particular LPG has run with *-n 50* option and, because it is a non deterministic planner, the shown results are the median values of the results obtained by 50 runs.

Table 11 Results for plan quality on *Transport* domain

Problem	FF(h_a)	Lama	LPG min	LPG median	ACO(FLA) min	ACO(FLA) median	Gamer
tra01	54	54	54	54	54	54	54
tra02	182	131	131	131	131	131	131
tra03	397	250	250	250	250	250	250
tra04	492	318	318	320	318	327.5	318
tra05	620	335	377	401	357	360	–
tra06	670	523	484	563.5	541	580.5	–
tra07	1079	528	519	612	682	1012	–
tra08	1225	682	811	887.5	1263	1318	–
tra09	1494	598	791	930	1174	1215.5	–
tra10	1705	762	2225	2225	2023	2023	–
tra11	456	456	456	456	456	456	456
tra12	886	594	594	594	594	594	594
tra13	1105	550	550	550.5	550	550	550
tra14	1507	1028	1055	1115.5	1032	1039	–
tra15	1963	1354	1072	1149	1384	1487.5	–
tra16	2937	1603	1717	1797	2766	2815	–
tra17	2349	2748	2410	2918	3485	3897	–
tra18	3779	2936	2834	3076	4757	4757	–
tra19	4450	4448	4227	4420.5	–	–	–
tra20	–	3877	–	–	–	–	–
tra21	478	478	478	478	478	478	478
tra22	632	632	632	632	632	632	632
tra23	630	630	630	630	630	630	630
tra24	1207	614	614	616	614	616.5	614
tra25	1716	1005	979	1051	1003	1004	–
tra26	1271	959	1026	1059	1004	1021.5	–
tra27	2517	962	1090	1108.5	1385	1402.5	–
tra28	4790	1794	2139	2336.5	3677	3723.5	–
tra29	1827	1505	1796	2014	2344	2531	–
tra30	3561	1553	2303	2435	2414	2414	–

The results relative to the plan quality are synthesized in Table 4 that shows, for each planner and for each domain, the two values *Solved* and *BQ* representing, respectively, the number of solved problem (at least in one run) and the number of problems for which a solution with the optimal (or the best-known value) cost has been found.

These results have been also analyzed with the χ^2 test and the results have been shown in Table 5 using the same technique previously presented for Table 2. From these data we can conclude that ACO(FLA) and Lama perform significantly better than both LPG and FF(h_a), and that LPG performs significantly better than FF(h_a).

Finally we have collected and analyzed the solution cost of the all attempted problem instances by the all tested planners. These data are analyzed by the Wilcoxon Signed-Rank test using the same technique and notations previously presented for Table 3; the results are shown in Table 6 while the data are extensively presented in Tables 7, 8, 9, 10, 11 and 12. For the deterministic planners Lama, FF(h_a) and Gamer the exact solution cost is reported, while for ACOPlan and LPG, which adopt stochastic algorithm in their search process, both the minimum cost they found

Table 12 Plan quality on *Woodworking* domain

Problem	FF(h_a)	Lama	LPG min	LPG median	ACO(AA) min	ACO(AA) median	Gamer
wood01	240	170	170	170	170	170	170
wood02	240	185	185	185	185	185	185
wood03	415	295	275	275	275	282	275
wood04	380	315	280	280	280	287.7	280
wood05	365	320	270	270	270	275.5	–
wood06	550	505	440	440	430	493.7	–
wood07	550	420	400	400	475	497.5	–
wood08	610	620	450	450	525	585.5	–
wood09	700	575	500	500	600	740.7	–
wood10	890	765	585	585	800	1012.9	–
wood11	145	130	130	130	130	130	130
wood12	310	225	225	225	225	225	225
wood13	265	235	215	215	215	215	215
wood14	295	275	230	230	245	245	225
wood15	405	435	270	270	280	342.2	270
wood16	385	360	315	315	315	334.4	–
wood17	460	335	290	290	335	424.3	–
wood18	580	640	470	470	525	582.6	–
wood19	665	530	420	420	465	553.6	–
wood20	680	835	525	525	730	863.9	–
wood21	105	95	100	100	95	95	95
wood22	195	185	185	185	185	185	185
wood23	225	195	195	195	195	195	195
wood24	290	245	245	245	245	245.3	245
wood25	680	435	420	420	400	535.7	–
wood26	365	240	240	240	240	253.1	–
wood27	465	480	370	370	410	459.5	–
wood28	665	665	1235	1235	660	733.9	–
wood29	785	735	545	545	745	854.4	–
wood30	925	825	695	695	930	1118.7	–

and the median value over the 50 runs are reported. When the data are missing the planner was not able to solve the problem at hand.

These data are analyzed by the Wilcoxon Signed-Rank test using the same technique and notations previously presented for Table 3; the results are shown in Table 6.

From these data we can deduce that ACO(FLA) and Lama are comparable in terms of solution cost, LPG performs a little worse than them while FF(h_a) is in general the worst performing.

Combining the results of these two last analysis we can conclude that ACO(FLA) and Lama are comparable both in terms of solving and optimizing capabilities; LPG is performing worse specially from the optimizing capability (i.e., it finds in general good solutions but it often fails to find the best possible solution); FF(h_a) is the worst performing both in terms of solving and optimizing capabilities with very relevant differences.

6 Conclusions

ACOPlan, an optimization planning algorithm has been presented. ACOPlan employs a colony of stochastic and heuristic–based ants–planners in the framework of ACO metaheuristic, an action cost based heuristic function FFAC and different pheromone models to optimize planning with respect to action costs.

Results of systematics experiments with benchmark domains suggest that, in general, action based pheromone models, like *Action–Action*, outperform state based ones, while the proposed novel pheromone model based on *Fuzzy–Level–Action* components has been proved to be a promising and effective tradeoff between performance and space cost. These results hold both for problem solving and optimization capabilities of the planner.

The comparison of ACOPlan with state of the art planners shows that the stochastic ACO based approach is optimal in many hard problems, and is competitive with respect to the percentage of solved problems and from the point of view of distribution of solution quality, as shown in the box–plot diagrams in Section 5. This is strongly promising since the ACO approach to planning can still be extended in many directions.

The analysis of results allows a deeper understanding of the role of component models in the problem solving and optimization capability of ACOPlan, which suggests useful guidelines for further development of ACOPlan and, at some extent, in the broader context of ACO algorithms.

– In order to produce effective search strategies which *distinguish a useful situation*, the components should capture the context in which an action is applicable, or a decision is taken (*maximum context details*);
– The component should not proliferate too much, for performance reason, and more importantly in order to *allow the pheromone to be deposited* on frequently used components. In other words, the components should not be too detailed in order to avoid to spread the pheromone on too many components which seldom will be found again in the search space (*minimum context details*)
– The model components should include *elements of the solution plans* (actions and precedence constraints, time steps or plan history etc.) instead of properties of

entities which can greatly differ in similar solution plans (e.g. a complete state description)
- The model components should verify the property that *two similar solution should produce similar components*, according to a given notion of similarity, i.e., a minimal perturbation on a solution producing a solution with the same optimality value should deposit a similar amount of pheromone on similar components.

This latter property has been shown to hold for AA and more strongly for FLA, allowing the pheromone to deposit on most of the components of perturbed solutions.

The promising behavior of FLA model will be experimented in future works in more general optimization contexts such as multiple costs and cost metrics involving resource consumption. In particular it is worth investigating extensions of the technique of *component fuzzification* introduced with FLA. According to this principle, given a metric distance D among components, a pheromone model can be fuzzified providing that: each component C which is similar to a given one C_{sol} appearing in a solution, will receive a pheromone amount in inverse proportion of the distance $D(C, C_{\mathrm{sol}})$. The technique has a general aim and it can also be applied to *state* based pheromone models, although these latter ones still present space complexity problems.

Another line of future research will be the study of ACOPlan with multiple heterogeneous pheromone models and/or colonies of heterogeneous ants. In multiple pheromone models the transition probabilities are computed by using the average of different pheromone types, while heterogeneous ants can use different search algorithms and/or heuristics. The purpose, in both cases, is to exploit the performance of different models/search methods and dynamically adapting them to the problem domain.

References

1. Bacchus, F., Kabanza, F.: Using temporal logics to express search control knowledge for planning. Artif. Intell. **116**, 123–191 (2000)
2. Baioletti, M., Milani, A., Poggioni, V., Rossi, F.: An ACO approach to planning. In: Proceedings of the 9th European Conference on Evolutionary Computation in Combinatorial Optimisation, EVOCOP 2009 (2009)
3. Baioletti, M., Milani, A., Poggioni, V., Rossi, F.: Ant search strategies for planning optimization. In: Proceedings of the International Conference on Planning and Scheduling, ICAPS 2009 (2009)
4. Baioletti, M., Milani, A., Poggioni, V., Rossi, F.: Optimal planning with ACO. In: Proceedings of AI*IA 2009. LNCS, vol. 5883, pp. 212–221 (2009)
5. Baioletti, M., Milani, A., Poggioni, V., Rossi, F.: PlACO: Planning with Ants. In: Proceedings of The 22nd International FLAIRS Conference. AAAI (2009)
6. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artif. Intell. **90**, 281–300 (1997)
7. Blum, C.: Ant colony optimization: introduction and recent trends. Physics of Life Reviews **2**(4), 353–373 (2005)
8. Bonet, B., Geffner, H.: Planning as heuristic search. Artif. Intell. **129**(1–2), 5–33 (2001)
9. Bylander, T.: The computational complexity of propositional strips planning. Artif. Intell. **69** (1–2), 165–204 (1994)
10. Cialdea, M., Limongelli, C., Poggioni, V., Orlandini, A.: Linear temporal logic as an executable semantics for planning languages. J. Logic, Lang. Inf. **16**, 63–89 (2007)
11. Conover, W.: Practical Nonparametric Statistics. John Wiley & Sons (1999)

12. Do, M.B., Kambhampati, S.: Sapa: a multi-objective metric temporal planner. J. Artif. Intell. Res. (JAIR) **20**, 155–194 (2003)
13. Dorigo, M., Stuetzle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA, USA (2004)
14. Edelkamp, S., Kissmann, P.: Gamer: bridging planning and general game playing with symbolic search. In: Proceedings of IPC-6 Competition (2008)
15. Garcia, M.P., Oscar Montiel, O.C., Sepulveda, R., Melin, P.: Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. Appl. Soft Comput. **9**, 1102–1110 (2008)
16. Gerevini, A., Serina, I.: LPG: a planner based on local search for planning graphs. In: Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS'02). AAAI Press, Toulouse, France (2002)
17. Helmert, M.: The fast downward planning system. J. Artif. Intell. Res. (JAIR). **26**, 191–246 (2006)
18. Helmert, M., Do, M., Refanidis, I.: International Planning Competition IPC-2008. The Deterministic Part. http://ipc.icaps-conference.org/ (2008)
19. Hoffmann, J., Nebel, B.: The FF planning system: fast plan generation through heuristic search. J. Artif. Intell. Res. (JAIR) **14**, 253–302 (2001)
20. Kautz, H., McAllester, D., Selman, B.: Encoding plans in propositional logic. In: Proceedings of KR-96, Cambridge, Massachusetts, USA (1996)
21. Kautz, H., Selman, B.: Unifying sat-based and graph-based planning. In: Proceedings of IJCAI-99, Stockholm (1999)
22. Keyder, E., Geffner, H.: Heuristics for planning with action costs revisited. In: Proceedings of ECAI 2008, pp. 588–592 (2008)
23. Menkes van den Briel, T.V., Kambhampati, S.: Loosely coupled formulations for automated planning: an integer programming perspective. J. Artif. Intell. Res. (JAIR) **31**, 217–257 (2007)
24. Nau, D., Ghallab, M., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann (2004)
25. Richter, S., Westphal, M.: The lama planner using landmark counting in heuristic search. In: Proc. of IPC-6 Competition (2008)
26. Rossi, F.: An aco approach to planning. Ph.D. thesis, Mathematics and Computer Science Dept., University of Perugia, Italy (2009)